FACE DETECTION FROM VIDEO STREAMING

by

Jorge Renato Torres García

Dissertation in Master in Data Analytics

Supervised by

Professor João Manuel Portela da Gama

Doutor Brais Cancela Barizo

2018

Biographic Note

Jorge Renato Torres Garcia was born on May 14th, 1991, in Lima, Peru. He finished his studies in Mechanical Engineering in 2012. In 2016, he was awarded an Erasmus Mundus scholarship in Data Analytics Master at the University of Porto.

Acknowledgments

I would like to thank the Erasmus Mundus Program for giving me the opportunity to attend the Data Analytics Master at the University of Porto.

Resumo

O objetivo deste trabalho é avaliar diferentes algoritmos para detecção de caras. Quatro estruturas diferentes foram testadas na sua qualidade e velocidade de detecção de caras num conjunto de imagens. Quadros por segundo e recall foram as principais métricas utilizadas. Todos os 4 métodos avaliados são implementações de open-sourced de publicações científicas. Os métodos avaliados foram os seguintes: um algoritmo de Histograma de Gradientes, uma rede convolucional neural simple, uma arquitetura em cascata chamada redes convolucionais em cascata multitarefa e uma estrutura de detecção de objetos chamada detector de disparo único. É demonstrado que os métodos de aprendizagem profunda são mais rápidos e precisos. Além disso, o feed de dados de treinamento para o modelo é crucial para um desempenho ideal de qualquer modelo. Conclui-se que, com hardware de baixo custo, é possível obter detecção de cara em tempo real num ambiente de vídeo de vigilância. O melhor modelo que foi testado, o MTCNN, alcançou quase 10 FPS em resolução de 400 x 380.

Abstract

The goal of this work is to evaluate the different frameworks for face detection. Four different frameworks were tested on their quality and speed of detecting faces in a set of images. Frames per second and recall were the main metrics used. All of the 4 methods evaluated, are open-sourced implementations of publicly available research papers. The methods assessed were the following: a Histogram of Gradients algorithm, a simple Convolutional Neural Network, a cascade type architecture called Multi-task Cascaded Convolutional Networks and a object detection framework called Single Shot Detector. It is demonstrated that deep learning methods are faster and more accurate. Also, the training data feed to the model is crucial for an optimum performance of any model. It is concluded that with inexpensive hardware is possible to achieve real time face detection on a surveillance video environment. The best model that was tested, the MTCNN, achieved almost 10 FPS in 400 x 380 resolution.

Index

Figure Index

Table Index

Acronyms

AFLW Annotated Facial Landmarks in the Wild

CNN Convolutional Neural Network

FDDB Face Detection Data Set and Benchmark

FPS Frames per second

GPU Graphics Processing Unit

HD High Definition

HOG Histogram of Gradients

MMOD Max-Margin Object Detection

MTCNN Multi-task Cascaded Convolutional Networks

ReLU Rectified linear unit

S3FD Single Shot Scale-invariant Face Detector

SSD Single Shot Multi-box Detector

SVM Support Vector Machines

1 Introduction

Nowadays, artificial intelligence is a common topic in our society. It is presented as the next revolution that questions if machines are more "intelligent" than humans. Some people have already risen concerns about the dangers it delivers. However, this work is on the frame of one of the areas where artificial intelligence, more specifically deep learning, has proven to really be a breakthrough: computer vision.

Thanks to convolutional and other types of neural networks, deep learning methods are the state-of-art method in the area of computer vision: the autonomous driving cars are an example where these methods are "changing the game". Despite the fact that this technology is on daily talk, its development and use is "prohibited" for most corporations: only the "Big 5" (Amazon, Apple, Facebook, Google, Microsoft) have the budget to do this. For example, Stanford University states that *"It would cost over $5 billion USD in hardware alone to analyze all the CCTVs in the UK in real time [with state-of-art algorithms]"* ("NoScope: 1000x Faster Deep Learning Queries over Video · Stanford DAWN," 2017).

Moreover, the recent news of China's surveillance program that uses artificial intelligence to score its population ("The Guardian view on surveillance in China: Big Brother is watching | Editorial | Opinion | The Guardian," 2017) has been given a lot of attention due to its science fiction nature, by saying the less. In this context, the development of a framework for facial recognition from video streaming should contribute, not only for "orwellians" purposes, but also to security, crime prevention and other areas. This work has the utopian dream to democratize artificial intelligence, by making it accessible to everyone, not only commercially, but also intellectually.

2 Face detection

In this chapter, it is explained the facial detection problem. First, common problems in the area of computer vision are presented in order to frame what detection and localization mean. Then, the relation between image and video is analyzed. Finally, a work proposal is given.

2.1 Computer vision problems

The main problems in the field of computer vision can be classified in four categories: classification, localization, instance segmentation and object detection. They differ on the objectives that have to be met. In the next paragraphs, a brief description of each one is given.

In an image classification problem, an image is ought to be classified into one of many different categories. In other words, the result of a classification problem is the category of which the algorithm predicts the image belongs.

The localization problems are meant to find the location of an object inside an image. Given an image and a set of objects, the algorithm in a localization problem should find where these objects are located. Most of the times, the result is presented in a rectangular box called bounding box.

Instance segmentation could be regarded as a more specific localization. In this type of problem, pixel by pixel mask of each of the detected objects should be found, not only a bounding box.

Object detection could be described as the task to correctly localize and classify all objects in an image that belong to a particular class. Based on that perception, object detection could be seen as a more general case than object location: in location problems, the numbers of categories are already defined; on detection, this information should be discovered (Farrugia, 2012).

2.2 Video as input source

Video is a sequence of images that our brain sees as a continuous moving picture. Our brain can make this moving picture idea with just 16 frames per second (fps) shown. In cinemas, video is shown at a rate of 24 fps. In television broadcasting, exists different standards for the frame rate: 30 for NSTC (North America, Japan) and 24 for PAL (Europe, Africa, SouthEast Asia) ("A Beginners Guide to Frame Rates : Aframe," n.d.)

More than half of the surveillance cameras on January 2016 use a 6-10 fps shooting (Ward, 2014). There are two basic reasons for that. The first one is that 6 frames per second is the minimum necessary frame rate in order to record a face of someone that is running (Ward, 2014). The other refers to the size of the video it is recorded: while using one quarter of the images per second (comparing with 24 fps), the size of the video is reduced by the same quantity without affecting the purpose to capture the face of the individuals that appear.

However, the most sold camera in Amazon.com has a rate of 25 fps and a quality of 720p ("Amazon.com: Tenvis HD Wireless IP Camera, Two-way Audio, Night Vision, 2.4GHz & 720P for Pet Baby Monitor, Home Security Motion Detection with Micro SD Card Slot (WH-TH661): Computers & Accessories," n.d.). This camera is for home use. Every camera has a bandwidth that depends on the quality of the frame and its rate, and also, in a shorter range, of the compression codec. So, in order to maintain the bandwidth restriction inherited to every purpose, it is possible to play with these parameters. For the purpose of detection, one may infer that a low frame rate with a high quality aspect should be the minimum necessary.

2.3 Work proposal

The goal of this work is to evaluate different frameworks for face detection. A benchmark specifying the methods and scores used to compare the different frameworks will be used. It is to remark that in Section 2.1, it is specified that detection, besides involving localization of the face in the image, it first demands finding out the total numbers of faces in the image, which can be zero.

It is also imperative to evaluate the performance of the frameworks in a real world environment. For this, two objectives will be tested: the framework should consume the minimum computer resources possible, so it can run in real time in an inexpensive consumer-final hardware. Moreover, the trade-off between precision and recall should also be analyzed.

3 Literature review

In this chapter, we make a review of the literature on this subject. First, the subject of object detection in images is exposed. The exploitation of temporal dependencies by using methods target to video is pointed out afterwards. To finalize, task-specific algorithms are mentioned.

3.1 Object detection on images

There is one algorithm that until this day is extremely popular due to its speed: the Viola/Jones face detector (Viola & Jones, 2001). Viola uses Haar wavelets in order to find regions in the images that can be used to recognize a face. The two most representative parts of a face, according to that algorithm, are the nose and the eyes (Fig. 3.1). The training is slow, as it requires multiple steps, but the detection is very fast. This algorithm is what is actually used in most consumer-devices. Two of the caveats of this algorithm, it that it needs the face to be align to the camera and the face should not have strange objects as glasses or hats. This work has also been successfully adapted on the detection of other objects besides faces.

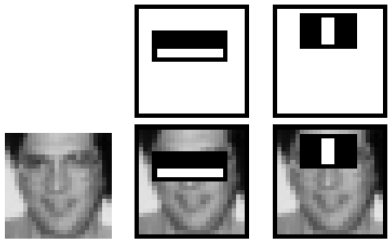

Fig. 3.1. The first two relevant features for Viola's algorithm (Viola & Jones, 2001).

In the literature exist a lot of algorithms that try to improve what Viola did. Dalal uses a technique called Histogram of Oriented Gradients (HOG) (Dalal & Triggs, 2005). HOG is a feature descriptor: a representation of an image that simplifies it by extracting useful information. *The basic idea is that local object appearance and shape can often*

be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions (Dalal & Triggs, 2005).

This technique detects semi-rigid objects in images by finding how dark is every pixel compared to the pixels surrounding it. Then an arrow is drawn by showing the direction in which the image is getting darker. These arrows are called gradients and they show the flow from light to dark through the image.

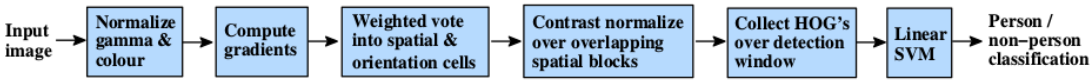

Fig. 3.2. HOG feature extraction and object detection chain (Dalal & Triggs, 2005).

In Fig. 3.2, the steps for training the algorithm are shown in more detail. This method is based on the evaluation of local histograms, which are well-normalized, of image gradient orientations in a dense grid. The steps can be grouped in two main parts: the feature extraction for each image and the object detection phased.

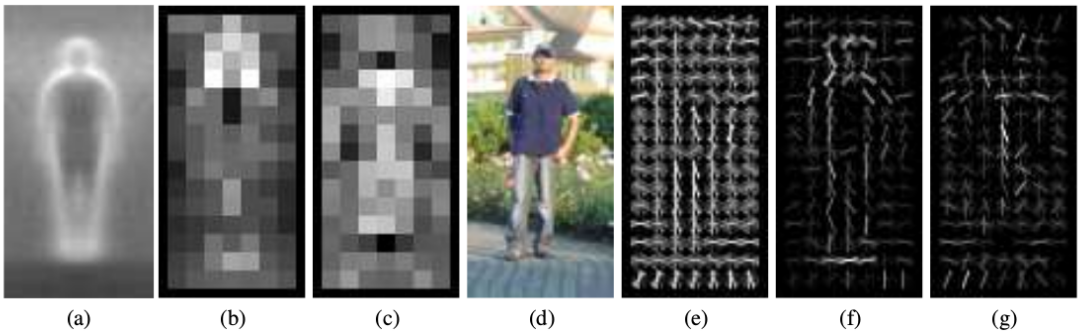

Fig. 3.3. HOG detection workflow (Dalal & Triggs, 2005).

In Fig. 3.3, the steps of the Histogram of Gradients algorithm are presented. In section (a), the average gradient image obtained over the training examples for human detection is presented. In section (b), it is shown the maximum positive weight that the Support Vector Machine algorithm gives to each pixel. In section (c), the negatives SVM

weights are shown. In (d), a test image is shown to explained how the algorithm will treat it. In (e), the HOG descriptor is compute. On the last two images it is shown the output of weighting the HOG descriptor respectively with the positive and the negative SVM weights.

Moreover, in the deep learning field, YOLOv2 (You Only Look Once) object detector (Redmon & Farhadi, 2016) is known to have beaten all scores in 2017. It is the second version of a popular algorithm for object detection called YOLO9000. This algorithm first divides the image in regions. Then, a neural network predicts the probability for belonging to a class; in our case, if it is a face or not. Finally, by using non-maxima suppression and thresholding, boxes that are not a valid detection are filter out. In Fig. 3.4, the process is explained. The algorithm only needs the image to be input once, reducing its computational time comparing with other neural networks approach. YOLOv2 achieves real time performance, 40-90 FPS in a Titan X GPU (Redmon & Farhadi, 2016). However, a GPU that costs more than 1000 Euros is needed for this("NVIDIA TITAN X Graphics Card with Pascal | GeForce," n.d.), a very expensive hardware if it is taken into account that is only for one video stream.

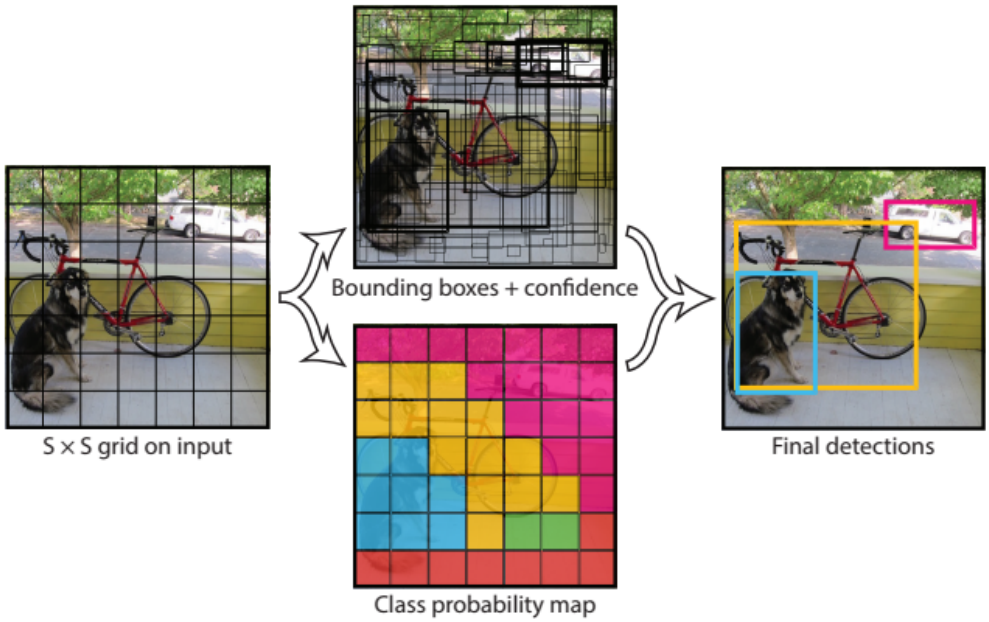

Fig. 3.4. YOLO prediction steps (Redmon, Divvala, Girshick, & Farhadi, 2015).

More deep learning techniques for object detection are also available. Single shot multi-box detector (Liu et al., 2016), also known as SSD, is a method for detecting objects in images using a single deep neural network. Between object detector frameworks that use neural networks is the fastest as it has a very simple architecture that takes advantage of predicting category scores and box offsets for a fixed set of bounding boxes using small convolutional filters applied to feature maps. The default model uses a VGG-16 model (Simonyan & Zisserman, 2014) as backbone.

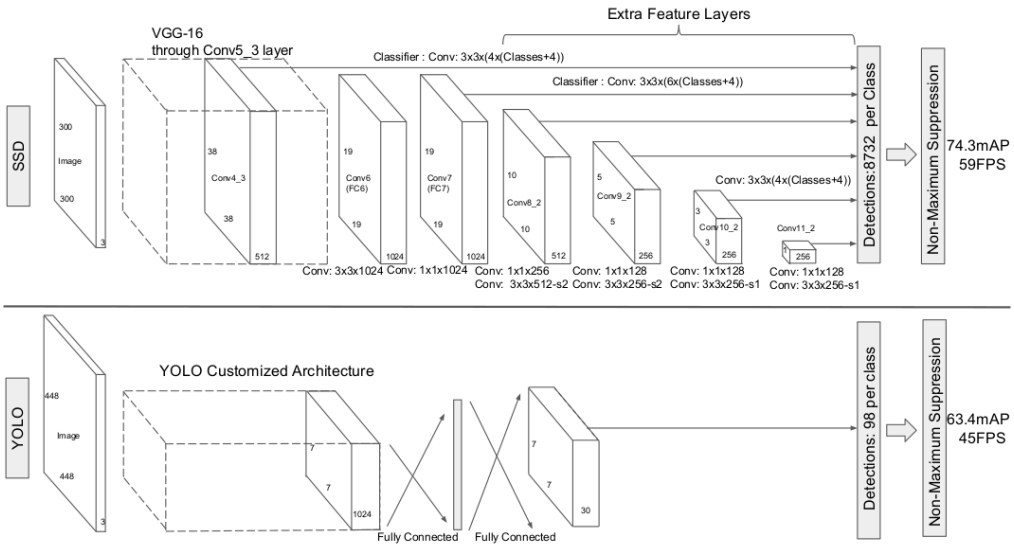

Fig. 3.5. SSD versus YOLO architecture (Liu et al., 2016).

In Fig. 3.5, both the architecture of the YOLO and SSD framework are presented. The better performance of the SSD framework can be explained by the use of several extra feature layers at the end of the base network. These layers help predict the offsets of the default prediction boxes for different scales, aspect ratios and their respective confidences. Moreover, despite the YOLO network being smaller, the fully connected layers greatly reduce the computation time of the network.

3.2 Object detection on video

As it is already mentioned, a video stream is just a sequence of images. NoScope (Kang, Emmons, Abuzaid, Bailis, & Zaharia, 2017) is a deep learning framework that takes advantage of the temporal relation between the frames of a video. NoScope automatically trains a sequence, or cascade, of models that preserves the accuracy of the reference network. By exploiting scene-specific locality with specialized models and temporal locality with difference detectors, this framework is computationally less expensive as it runs these cheaper models whenever is possible.

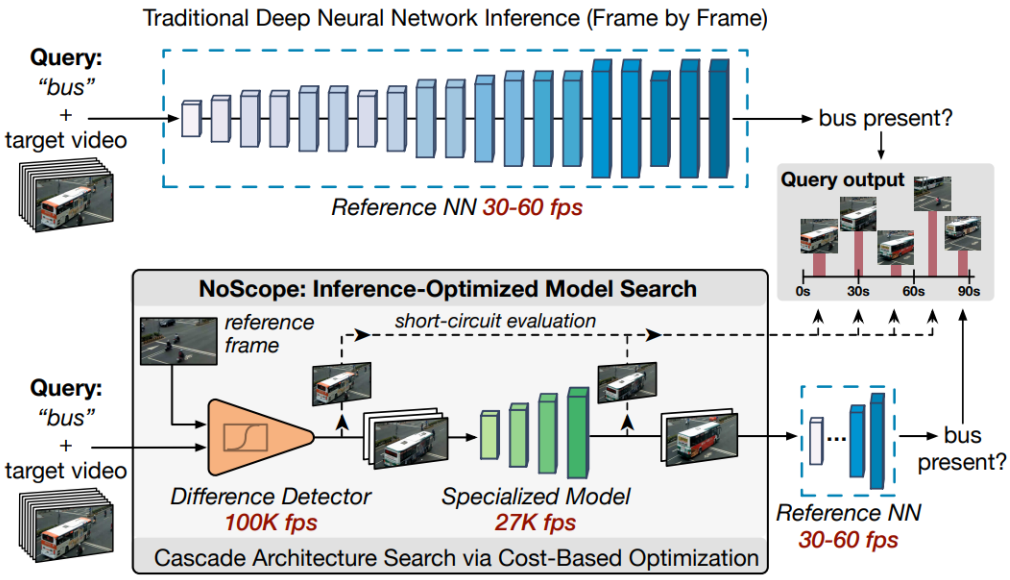

Fig. 3.6. NoScope Framework (Kang et al., 2017).

The structure of the NoScope Framework is introduced in Fig. 3.6. NoScope being given an input video, target object and reference neural network, e.g. YOLO network, *automatically searches for and trains a cascade of models—including difference detectors and specialized networks—that can reproduce the binarized outputs of the reference network with high accuracy—but up to three orders of magnitude faster* (Kang et al., 2017).

Another algorithm used for object detection in video is (Shafiee, Chywl, Li, & Wong, 2017), called Fast YOLO. This framework accelerates YOLOv2 to be able to perform object detection in video on embedded devices in a real-time manner. It uses a motion-adaptive inference that could be seen as having a key frame that persists in time.

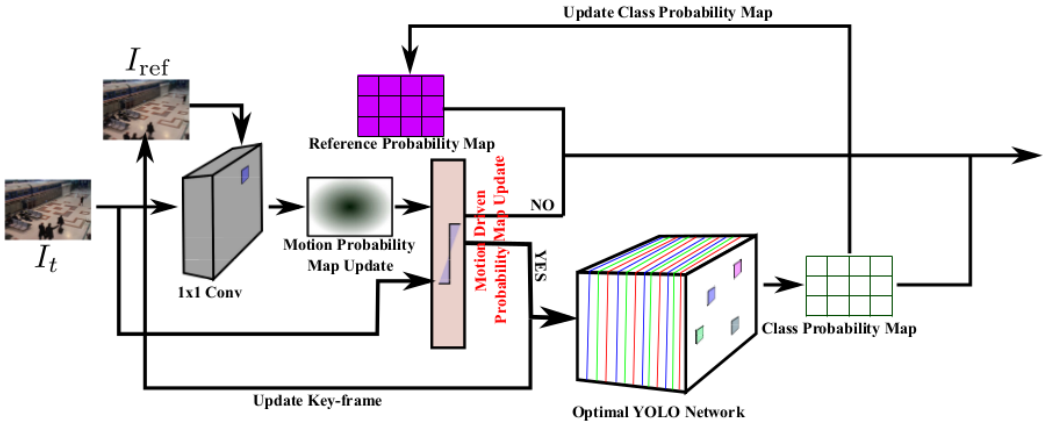

Fig. 3.7. Fast YOLO Framework (Shafiee et al., 2017).

In Fig. 3.7, the Fast YOLO Framework for object detection in video is presented. *For each video frame I_t , an image stack consisting of I_t and a reference video frame I_{ref} is passed into a 1×1 convolutional layer to compute a motion probability map. The motion probability map, along with I_t is then passed into a motion-adaptive inference module to decide if deep inference is needed to compute a class probability map (Shafiee et al., 2017)*. If deep inference is needed, the optimal YOLO network is used to computed an updated class probability map. I_t is now stored as I_{ref} and the reference probability map is also updated. Otherwise, the reference class probability map is used directly. In this way, the Fast YOLO framework does not need to run the deep inferences framework for every frame, which helps greatly reduce computation time.

3.3 Face detection

Deep learning algorithms have been known to not be task-specific: a convolutional neural network can be applied to any kind of image. Despite this, frameworks that are specific for face detection have been developed.

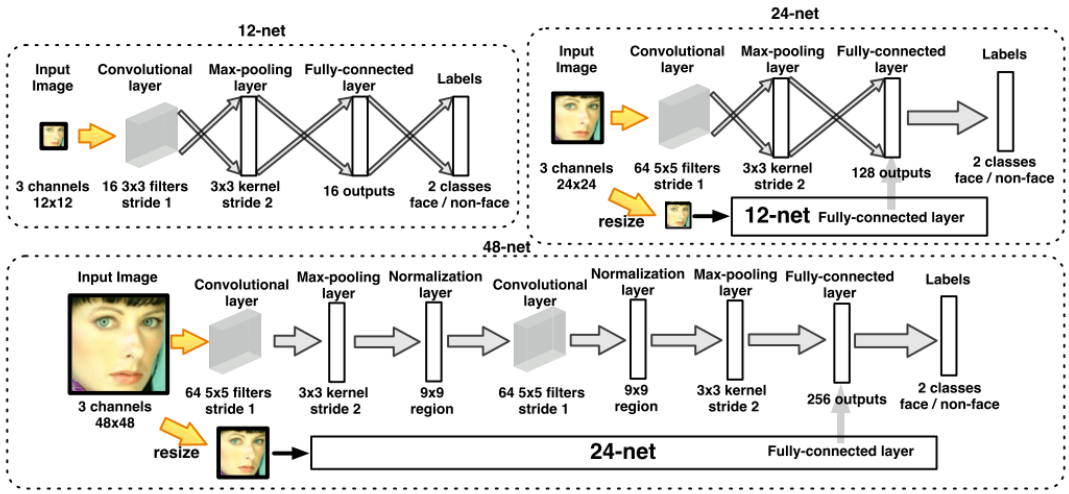

Fig. 3.8. Convolutional neural network cascade (Li, Haoxiang and Lin, Zhe and Shen, Xiaohui and Brandt, Jonathan and Hua, 2015).

Almost all of state-of-art algorithms specifically for face detection involves a convolutional neural network cascade. (Li, Haoxiang and Lin, Zhe and Shen, Xiaohui and Brandt, Jonathan and Hua, 2015) is one of the pioneers in this framework topology. In this algorithm (Fig. 3.8), 3 different neural networks are trained one after the other with different input size (12x12, 24x24, 48x48) of the same image. This is done in order to obtain bounding boxes from the lower resolution image, and then calibrate the best ones in the latter stages. This algorithm *detects faces from 640×480 VGA images (typical surveillance scenario) runs at 14 FPS on a single CPU core for VGA-resolution images and 100 FPS using a GPU* (Li, Haoxiang and Lin, Zhe and Shen, Xiaohui and Brandt, Jonathan and Hua, 2015).

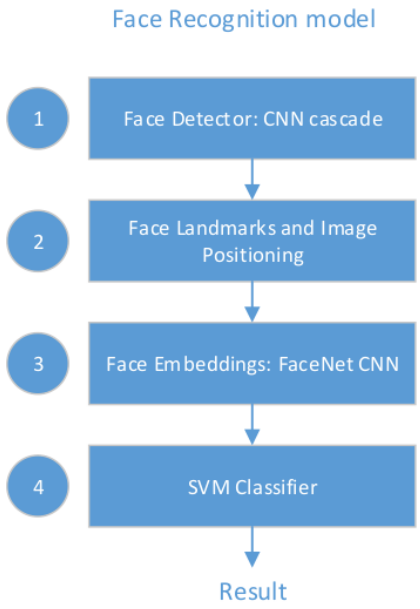

Fig. 3.9. Facetime framework (Arsenovic, Sladojevic, Anderla, & Stefanovic, 2017).

In another study called Facetime (Arsenovic et al., 2017), this same convolution neural network cascade is used as face detection step in a more complete framework. In Fig. 3.9, the Facetime framework is shown. First, the previously described cascade algorithm of Li is used for face detection. After the face is encountered, the face landmarks are found so the face can be aligned. Then, with the face already aligned, the face embeddings are obtained. This is a 128-dimension feature map that characterizes each different face. At the end a SVM classifier is used to whom the face belongs. Facetime was successfully applied in a real environment achieving a 95% accuracy with a low-capacity hardware and a standard IP camera.

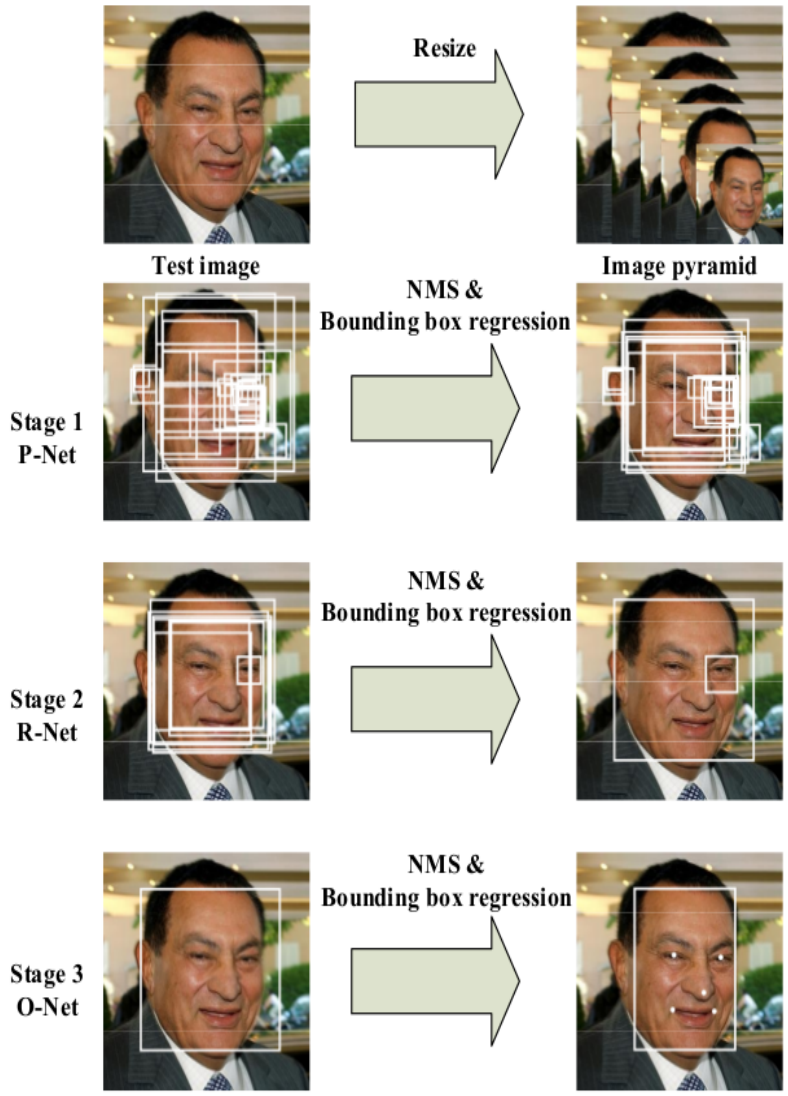

Fig. 3.10. Multi-task Cascaded Convolutional Networks (K. Zhang et al., 2016).

(K. Zhang et al., 2016) also makes use of a cascade type architecture achieving state-of-art result. Zhang framework called Multi-task Cascaded Convolutional Networks, consists on resizing the image into different scales to build an image pyramid. This will be the input for the three-stage cascaded framework (Fig. 3.10). In the first stage, candidate facial windows are obtained. In the second stage, all candidates are fed to another convolutional neural network which further rejects a big number of candidates and calibrates them. The final stage is similar to the second one, but in the latter one, the network will output five facial landmarks positions also.

(S. Zhang et al., 2017) is based on the SSD object detector framework which was previously mentioned. Zhang overcomes the difficulty of the SSD framework to detect small faces by developing an architecture with a wide range of anchor-associated layers, whose stride size gradually double from 4 to 128 pixels. The new framework is known as Single Shot Scale-invariant Face Detector, S3FD.

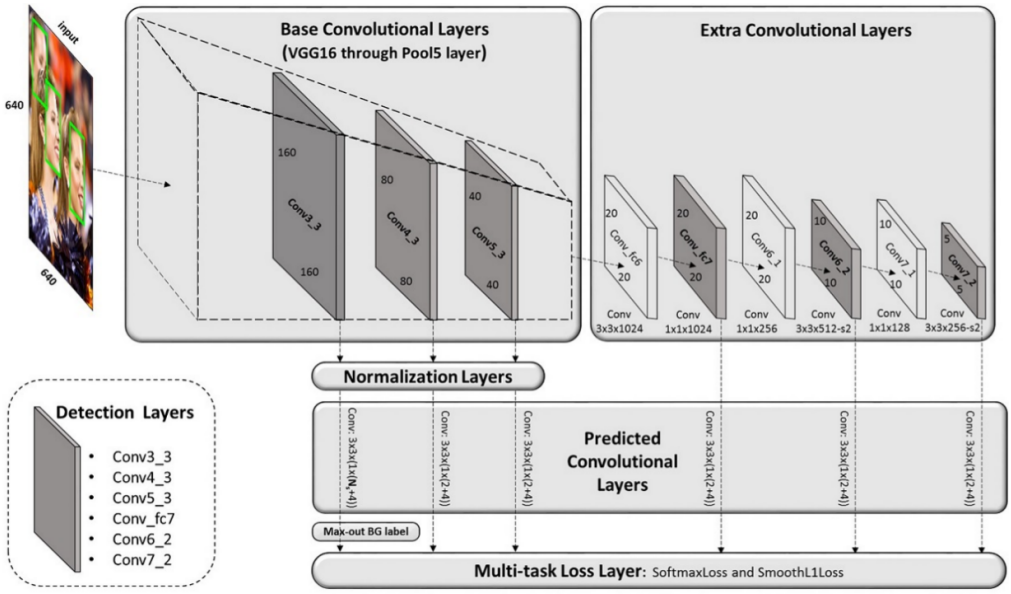

Fig. 3.11. S3FD architecture (S. Zhang et al., 2017).

In Fig. 3.11, the architecture of the S3FD framework is presented. As previously stated, the extra convolutional layers help the network find small objects. In Fig 3.12, for example, an image with 1000 faces is presented. The S3FD identifies 853 of them. The colorbar on the right of the image expressed the detection confidence. As it can be seen from the image, S3FD is robust to pose, occlusion, expression, makeup, illumination, blur and face size.

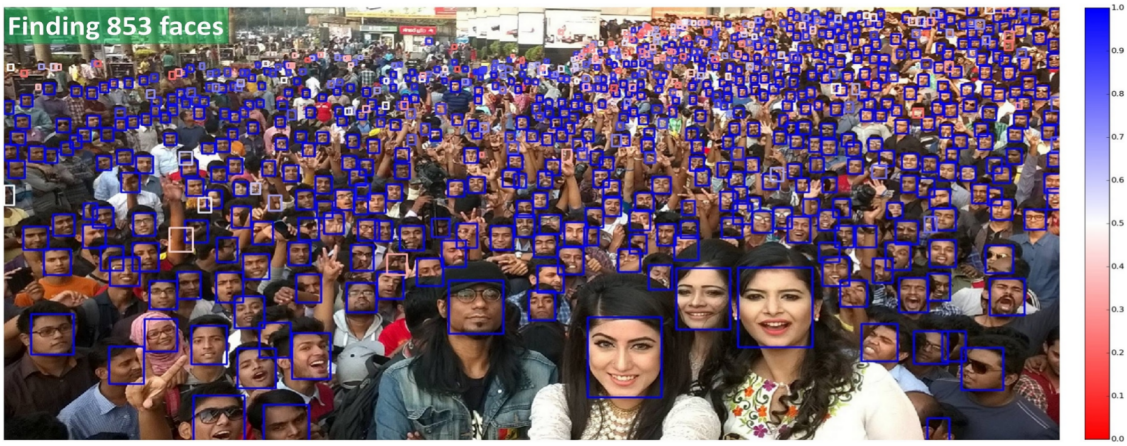

Fig. 3.12. S3FD detection example (S. Zhang et al., 2017).

Moreover, one more algorithm (Kalinovskii & Spitsyn, 2015) proved that is possible to achieve real-time performance in inexpensive devices. This algorithm claims that it can process 4K Ultra HD video stream in real time (up to 27 fps) on mobile platforms (Intel Ivy Bridge CPUs and Nvidia Kepler GPUs). It finds faces with the dimension of 60×60 pixels or higher.

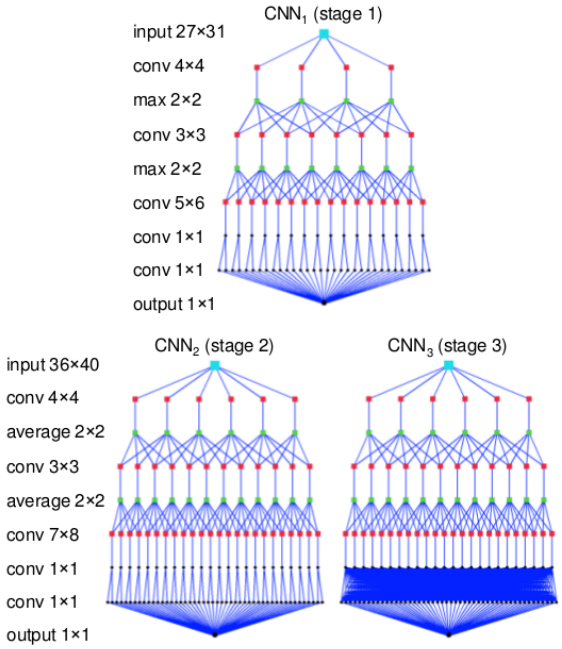

Fig. 3.13. CNN structures (Kalinovskii & Spitsyn, 2015).

As is seeing on Fig 3.13, this algorithm also uses a Cascaded type framework. The main difference with the other ones is that the last two neural network of the cascaded run asynchronously (Fig. 3.14). With the proper hardware configuration, this asynchronous framework can really speed up processing time.

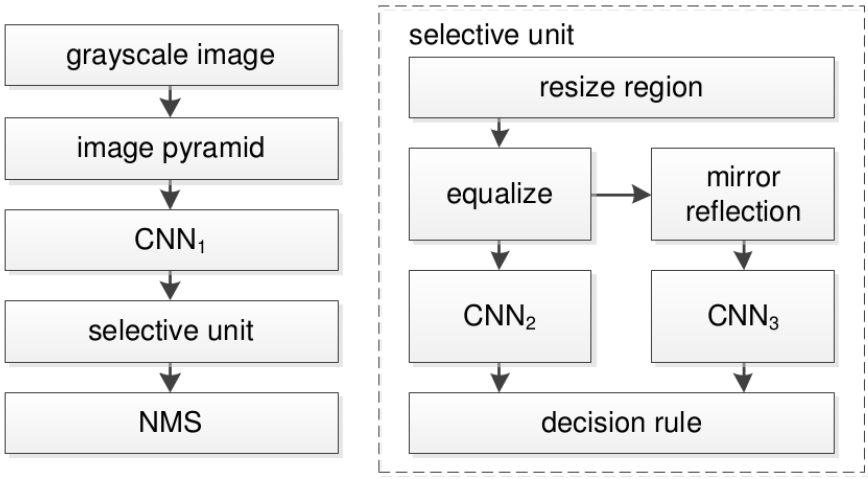

Fig. 3.14. Detector design (Kalinovskii & Spitsyn, 2015).

4 Methodology

In this section, the methodology for benchmarking different frameworks will be presented. This section is divided as follows: the algorithms to be tested, the benchmarking protocol including the data and the metrics that will be used, and finally the software that will be employed.

Is important to state that, until this date, for the author, there is not a standard benchmark for video processing as is a relative new field and in order for the community to agree in one it naturally takes time. For face detection in image, there is one well-established benchmark called Face Detection Data set and Benchmark (Jain & Learned-Miller, 2010), also known as FDDB. This benchmark will be used, with the addition of the metric about processing time in order to benchmark how these algorithms will run in a video environment.

4.1 Frameworks

ID	Library	Training data	Notes
HOG	Dlib	3000 images from the ALFW	MMOD as loss function
CNN	Dlib	6975 faces from ImageNet, AFLW, and others	MMOD as loss function
MTCNN	Tensorflow	MS-Celeb-1M dataset	
SSD	Caffe	Some huge and available online dataset	Resnet-10 like architecture as backbone
S3FD	Caffe	12880 images of the WIDER FACE training set	

Table 4.1. Frameworks to benchmark

Five frameworks for face detection were selected for benchmarking. These frameworks were chosen in order to have a diverse set of frameworks: from the simplest algorithm that is used for more than 10 years ago, like the Histogram of Gradients until the state-of-art deep learning method called Single Shot Scale-invariant Face Detector. Open data sets were used for training all these frameworks. It is imperative to point out that the FDDB face dataset was not included in any of the training sets in neither of the algorithms as is a requirement in order to avoid over fitting.

The simplest framework is based on the Histogram of Gradients (HOG) that was already described. In particular, this HOG framework uses a method called Max-Margin

Object Detection (King, 2015) for classifying and suppressing sub-windows detections of an image. In contrast to the common step of binary classifying and then applying a non-maximum suppression step, the MMOD method does not perform any sub-sampling and optimizes over all sub-windows giving very impressive results. This framework is implemented using the Dlib library (King, 2009). For this framework the Annotated Facial Landmarks in the Wild (AFLW) (Köstinger, Wohlhart, Roth, & Bischof, 2011) was chosen. From the 25 thousand annotated faces that this dataset contain, only 3000 images were used for training this algorithm.

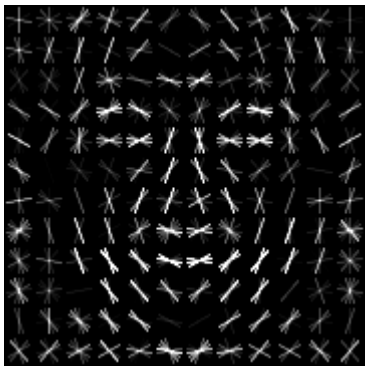

Fig. 4.1. HOG descriptor for a face (King, n.d.-b).

For example, in Fig. 4.1, it is shown the learned HOG descriptor from 4 images containing 18 faces. It surprisingly looks like a face with only these very small sample of images. It is known that this algorithm does not need a big quantity of images for training contrary to deep learning algorithms. However, equally the implementation of the HOG used in this work it is trained with 3000 faces as the training only took 3 minutes.

Another thing that this framework had in particular is the used of the Max-Margin Object Detection instead of the SVM that the original paper uses. In contrast to the classifying and suppressing steps of the windows, this method learns a regression: it optimizes by running a regression between all the sub-window detections. Moreover, the MMOD uses all the sub-windows of an image as training. That is another reason why the HOG framework in this case required less than 20 images.

The second framework to be evaluate dis a simple Convolutional Neural Network (CNN) with the previous MMOD as loss function. The CNN framework consist of a convolutional neural network with some layers for down sampling follow by a fully connected network with ReLU function. Batch normalization is also used. This framework is also developed using the Dlib library. This framework was trained with 6975 faces from the same dataset, but also from other datasets like ImageNet (Jia Deng et al., 2009).

The architecture of this Convolutional Neural Network is introduced. First, two convolutional layers are defined: one with 5x5 filter and another with 3x3. A downsampling of 8 times is achieved by the use of these layers concurrently for each of the three channels of our image. ReLU and batch normalization are also used. With these, a feature map of 32 dimensions is obtained. These features are then passed by 4 more convolutional layers of 3x3. The last layer of this four has only one channel. The values of this last channel are large when the network found an object at a particular location.

The model is trained with a batch size of 150. The starting learning rate is 0.1. If in 3000 iterations the MMOD loss has not had an improvement it will reduce the learning rate (King, n.d.-a). The training is stop when the learning rate is less than 0.0001.

The third framework is an implementation of the Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks (K. Zhang et al., 2016) which have been previously presented (MTCNN). It is implemented in Tensorflow (Martin~Abadi et al., 2015) library. The MS-Celeb-1M dataset (Guo, Zhang, Hu, He, & Gao, 2016) was used by this framework as training set. This dataset is one of the hugest dataset in the face recognition domain as it counts with more than 10 million images with 100 thousand different faces.

The loss function of this model is a weight summed of three objective functions: a cross-entropy loss function for face/non-face classification, a euclidean loss for each candidate window and a euclidean loss for each facial landmark. The model was train for 16 epoch with a batch size of 384.

For the fourth framework, an implementation of the Single shot multi-box detector (Liu et al., 2016) previously described with a Resnet-10 like architecture as backbone is used. It is implemented in Caffe (Jia et al., 2014) library. This framework does not stated from which dataset the training data was obtained.

This implementation was trained for 14000 steps with 300 x 300 images. The loss function is a weighted sum of the location loss and the confidence loss. The location loss is a L1 loss between the predicted bounding box and the annotated box. The confidence loss is the softmax loss over if it is belong to class "face" or not.

The fifth framework is an implementation of previously described state-of-art method Single Shot Scale-invariant Face Detector (S. Zhang et al., 2017). It is implemented in Caffe library. 12880 images from the Wider Face dataset (Yang, Luo, Loy, & Tang, 2016) are used as training set for this model.

4.2 Benchmarking protocol

As previously mentioned, the Face Detection Data Set and Benchmark (FDDB) (Jain & Learned-Miller, 2010). This benchmark included a dataset of faces that will help to evaluate the frameworks. This dataset consists of 5171 faces in a set of 2845 images taken from the Faces in the Wild data set, is important to consider that is different from the Labeled Faces in the Wild dataset, as some frameworks that are going to be tested are trained with this dataset. It includes a wide range of difficulties including occlusions, difficult posses, low resolution and out-of-focus faces. Moreover, there is not restriction on the orientation of the head (pose-invariant).

The medium size of the image is 399 x 377 with a standard deviation of 62 x 64. The size of smallest image is 171 x 440 and the biggest image size is 450 x 450. There is no face smaller than 20 x 20. It includes both gray-scale and color images.

Fig. 4.2. FDDB sample images with face annotations (Jain & Learned-Miller, 2010).

The face annotations in the FDDB data set are ellipses as these reflected better the shape of the face, comparing with a rectangular one. As it can be seen in Fig. X, the faces are not always frontal nor in straight position. Also in the image from the right, the out of focus is distinguishable in the right face.

The face annotations are split into ten folds. Two experiments are proposed by the benchmark: a 10-fold cross-validation or unrestricted training. In the former, the model is trained with 90% of the data to predict the other 10%, this is repeat 10 times. In the unrestricted training data outside the FDDB data set is used as training set, which is the one used on this work.

Moreover, there are two ways to report the evaluation metrics: the continuous score and the discrete score. In the continuous score, the area of intersection between the detection and the annotated region in percentage is assign for each face. In the discrete score, if the area of intersection is more than 0.5, a score of 1 is assign for the prediction. In this work, the continuous score is not useful as is not relevant for a surveillance environment as is imperative to have the whole face for analysis.

So, each framework will be run with the FDDB as prediction set. A pre-processing step of normalizing the image will be run for all deep learning methods. For the HOG framework, the image will be converted into grayscale. The output of each algorithm will be the predictions of the localization of faces in a bounding box format. For the FDDB benchmark, the face should be report as follows:

<name of the image>

<number of faces>

<face X1>

<face X2>

The format of face X will be rectangular as it is the standard for all the algorithms. For rectangular regions, each face is represent by:

<left_x top_y width height detection_score>

With this information, the FDDB protocol will be run. First the ratio of intersection between the prediction area and the annotated face will be compute (Equation 4.1). Then, if this ratio is higher than 0.5, the face will be considered correctly predict for the discrete score. For the continuous score, this ratio will be considered for reporting the recall rate. In other words, the mean of the ratio of intersection for all faces will be reported in the continuous score.

$$\text{Ratio of intersection} = \frac{detection\ area \cap annotated\ area}{detection\ area \cup annotated\ area}$$

Equation 4.1. Ratio of intersection

Finally, the speed of the algorithm will be evaluated in frames per second. It measures how many images the algorithm can test in one second. Taking into account a surveillance environment, the minimum of 6 frames per second is required in this work.

For this, the speed in seconds for each image to undergo each framework will be measure. For converting seconds in Frames per second, equation 4.2 is used. Then the mean and standard deviation for each algorithm to predict the whole set of images will be reported.

$$Frames\ per\ second = \frac{1}{s}$$

Equation 4.2. From seconds to Frame per Seconds

4.3 Hardware

	Specifications
CPU	Intel® Core™ i5-4200U CPU @ 1.60GHz × 4
GPU	GeForce GT 740M/PCIe/SSE2 2GB (CUDA capability 3.0)
Memory	16 GB DDR3L
SSD	Samsung 850 EVO 500 GB
OS	Ubuntu 16.04 LTS 64-bit

Table 4.2. Hardware specifications

The hardware used for this work is my personal computer that was acquired more than 3 years ago. As stated before, how face recognition algorithms run in an inexpensive hardware is one of the main purposes of this research. The CPU is a 4th generation Intel processor with only 2 physical threads and a speed of 1.6 GHz that is relative slowly. The GPU is a CUDA (Nickolls, Buck, Garland, & Skadron, 2008) capable GPU with only 2 GB of memory. This will be our main bottleneck as deep learning algorithms are GPU intensive both in processing and in memory aspects. The RAM size is 16 GB which can be considered good for this application. Our hard drive is a Samsung Solid State Drive with 500 GB of space, which can also been considered good for this task. The operation system is an open source Linux distribution called Ubuntu, which is ideal for general data science applications.

4.4 Software

Python 3.6 is used for the development of the pipeline of the benchmark. All of the 5 methods are implemented in this language with the help of different libraries. OpenCV (Bradski, 2000) is a library for easily handling image and photos that will help us with the pre and post processing. Numpy is an optimized library for array manipulation. Caffe (Jia et al., 2014) deep leaning framework developed by Berkeley AI Research, Dlib (King, 2009) a modern C++ toolkit containing machine learning algorithms and Tensorflow (Martin~Abadi et al., 2015), Google open-sourced library for tensor manipulation, will be utilized.

5 Results and discussion

One of the limitations of our hardware was encountered on trying to load the frameworks. The state-of-art S3FD is too big to fit in the 2 GB memory of the GPU. Henceforth, only the benchmark of the remaining four frameworks is presented. All the frameworks were run with their default hyper parameters.

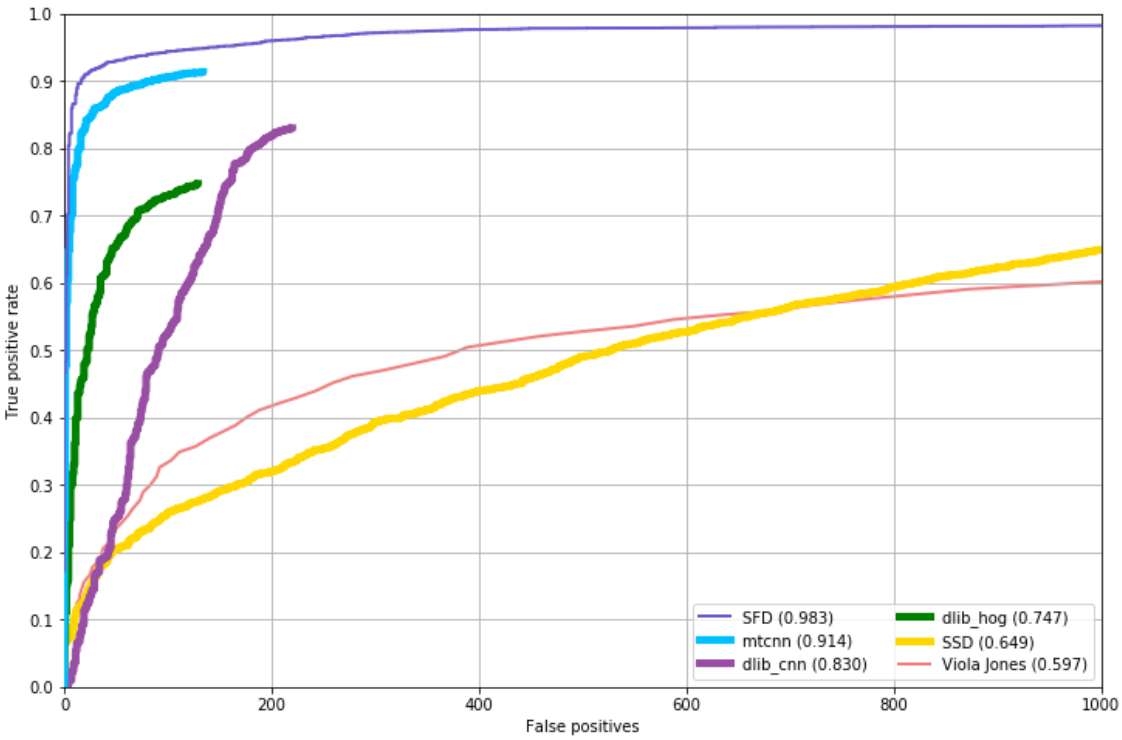

Fig. 5.1. True positive rate vs False positives, FDDB discrete score.

In Fig. 5.1, the results of the FDDB discrete score is shown for the four algorithms. The recall (True positive rate, from all the faces the algorithm predicts which ones are really faces) is plot against the number of False positives. As previously mentioned, in the discrete score, for a face to be considered correctly predict the ratio of intersection between the predict and the annotated bounding box should be more than 50%. For having an upper boundary, the S3FD is shown in a blue thin line and for the lower

boundary, the old-known Viola Jones face detector is shown in a red thin line. Both of these boundaries curves were obtained from the FDDB database.

Is very surprising that the SSD framework is as worst as the Viola Jones detector. The maximum True positive rate that the SSD achieved is 0.79 with a little less than 2400 False positives. The SSD model is the only one that did not specify with which data it was trained. It can be inferred from this result that the training data was of bad quality. It is also to be noted that varying the threshold of the prediction for this model (the default is 0.5) won't help overcome the False Positives, as most of the False positives have a score of more than 0.9.

The best model is the MTCNN, which between the three remaining it was expected to perform the better. It achieves a recall of 0.914, which is relative very good. In order to say which model is the second best, which have to take into account the task. For a surveillance environment, is better to have a higher recall sacrificing the rise of false positives, so one can retrieve all the faces in a scene. With this in mind the CNN framework achieve a better recall than its HOG counterpart, 0.83 versus 0.747. For being a very simple model, the HOG framework got a much better precision than the CNN, calling less false positives.

ID	FPS	Recall (1000 FP)
HOG	7,9 ± 1,0	0,747
CNN	4,3 ± 0.5	0,830
MTCNN	9,7 ± 3,4	0,914
SSD	13,7 ± 1,6	0,649

Table 5.1. FPS and Recall, FDDB benchmark

The only model that runs in the CPU is the HOG, with speed of 8 FPS it can be run in real time in a surveillance environment, which is really great taking into account is good precision metric. The CNN framework is the slowest of the 4 by running at 4.3 FPS. It shows that a non-optimum design of the neural network can spoil the speed. The MTCNN runs at almost 10 FPS, arising as the second fastest framework, only behind the SSD with 13 FPS. The 3.4 FPS of standard deviation (more than one third of the mean) is to be expected from the MTCNN. As a cascaded type architecture, if there is no face encounter in the first stage, the model won't run anymore steps. However, in environments with a lot of faces, the model will have its speed decrement.

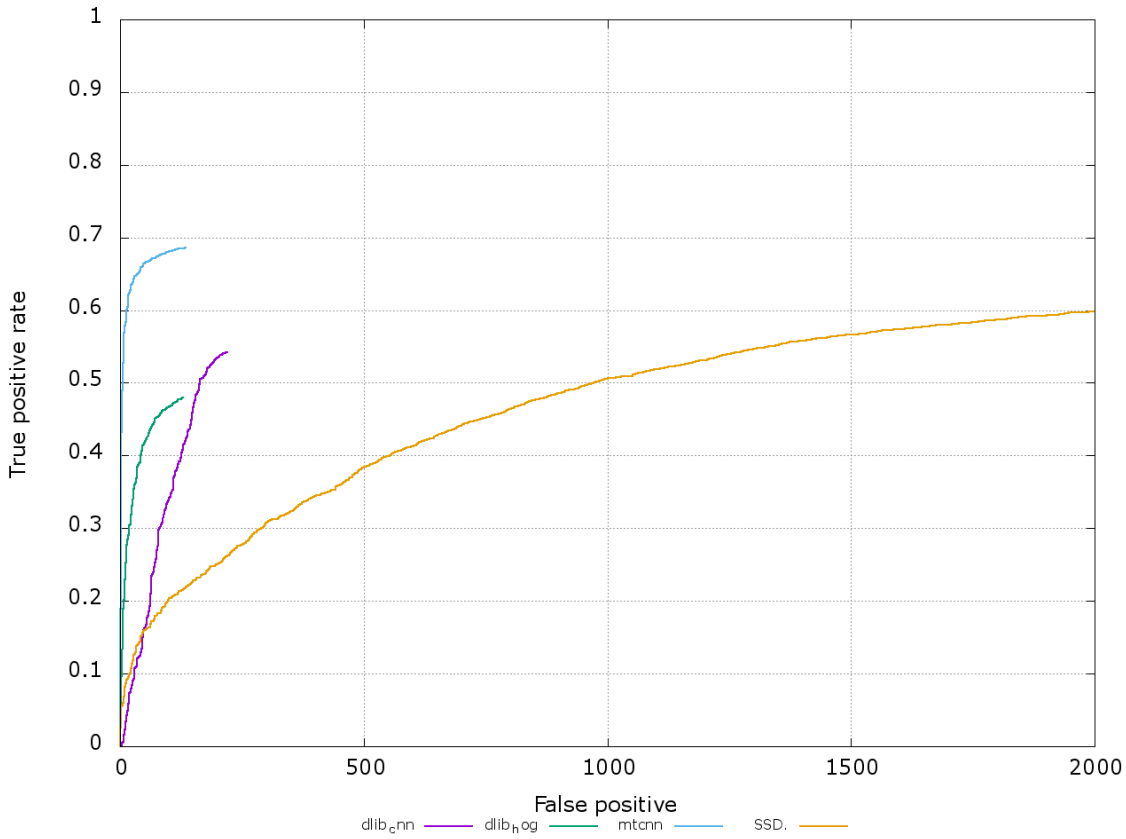

Fig. 5.2. True positive rate vs False positives, FDDB continuous score.

In order to have more lights of why the SSD performed really bad. One could hypothesized that may be because of the predicting bounding boxes been too small for getting more than the 50% area of overlapping need in the discrete score for computing as a correct prediction. In Fig. 5.2, the curve of continuous score is presented. As previously mentioned, this is the mean of the ratio of intersection for each face. The MTCNN framework continues to be the best and it can be inferred that is predicting bounding box are well align with the annotated ones. The SSD framework continues to perform very poorly, however, now in the limit of the False positives (2000) it has a better recall than the HOG and CNN framework. This signifies that even though the SSD had a lot of false positives, the predicting bounding box of the true positives is much more precise.

Fig. 5.3. CNN false alarms on FDDB dataset (King, n.d.-c).

In Fig. 5.3, the false alarms of the CNN framework are shown. This can give us more clues on when and why the false alarms appear. As it can being seen from the false positives, most of them are faces. However, is not easy when to determine when the oclussion is too much to consider that is not a face or if it is too blur, for example. In order to overcome these, the FDDB authors gathered a panel of experts for selecting the faces. One can easily think “Hey! But that is really a face!”. That is why in this work we strictly follow the FDDB protocol as everyone will have their own opinion on which are the “real” faces the algorithms must recognize.

All in all, the MTCNN framework appears as the winner in this work as it achieved the best precision, the best recall and second fastest FPS (only behind the SSD, which really performed bad in the FDDB test) framework of the four that were tested. Moreover, this

framework already gives as output five facial landmarks positions that will be really helpful for posterior analysis of the faces encounter. These facial landmarks can help us, for example, to align the faces, which is the next step in a face recognition task.

6 Conclusion

Five different frameworks were selected to be tested on their quality and speed of detecting faces in a set of images. Frames per second and recall were the main metrics used. The S3FD method, which is a state-of-art method for face detecting did not fit in our 2 GB GPU so at the end only 4 methods were tested. All of the 4 methods evaluated are open-sourced implementations of publicly available research papers. The methods assessed were the following: a Histogram of Gradients algorithm, a simple Convolutional Neural Network, a cascade type architecture called Multi-task Cascaded Convolutional Networks and a object detection framework called Single Shot Detector.

From our analysis, deep learning methods appear to be more accurate. The HOG framework is the only non deep learning method and performed the worst in recall. Moreover, all deep learning methods benefit of the capability to be run in the GPU achieving faster performance.

Referring to the results, what is worthwhile mentioning is the bad performance of the SSD framework. We could assume that this might happen because the training set is not as variate as the other ones, or maybe not suitable at all for the face recognition task. This highlights the importance of an appropriate data selection for training any machine learning model.

It can also be concluded that with inexpensive hardware (GT 740M), it is possible to achieve real time face detection on a surveillance video environment. The best model that was tested, the MTCNN, achieved almost 10 FPS in 400 x 380 resolution. The recall of this model was 91.4 which is really promising for continuing the development of it in a real working environment.

7 References

A Beginners Guide to Frame Rates : Aframe. (n.d.). Retrieved January 13, 2018, from http://aframe.com/blog/2013/07/a-beginners-guide-to-frame-rates/

Amazon.com: Tenvis HD Wireless IP Camera, Two-way Audio, Night Vision, 2.4GHz & 720P for Pet Baby Monitor, Home Security Motion Detection with Micro SD Card Slot (WH-TH661): Computers & Accessories. (n.d.). Retrieved January 14, 2018, from https://www.amazon.com/Wireless-Two-way-Security-Detection-WH-TH661/dp/B071DDBT7M/r

Arsenovic, M., Sladojevic, S., Anderla, A., & Stefanovic, D. (2017). FaceTime – Deep Learning Based Face Recognition Attendance System, 53–58.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *IEEE Conference on Computer Vision & Pattern Recognition*, 886–893.

Farrugia, R. A. (2012). *Multimedia Networking and Coding*. Information Science Reference. Retrieved from https://books.google.pt/books?id=uMKeBQAAQBAJ

Guo, Y., Zhang, L., Hu, Y., He, X., & Gao, J. (2016). MS-Celeb-1M: A Dataset and Benchmark for Large Scale Face Recognition. *CoRR*, *abs/1607.0*, 1–17. https://doi.org/1607.08221

Jain, V., & Learned-Miller, E. (2010). FDDB : A Benchmark for Face Detection in Unconstrained Settings, UM-CS-2010-009.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., … Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv Preprint arXiv:1408.5093*.

Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. https://doi.org/10.1109/CVPRW.2009.5206848

Kalinovskii, I., & Spitsyn, V. (2015). Compact Convolutional Neural Network Cascade for Face Detection. Retrieved from http://arxiv.org/abs/1508.01292

Kang, D., Emmons, J., Abuzaid, F., Bailis, P., & Zaharia, M. (2017). NoScope: Optimizing Neural Network Queries over Video at Scale. https://doi.org/10.475/123

King, D. E. (n.d.-a). dlib C++ Library: Automatic Learning Rate Scheduling That Really Works. Retrieved May 6, 2018, from http://blog.dlib.net/2018/02/automatic-learning-rate-scheduling-that.html

King, D. E. (n.d.-b). dlib C++ Library: Dlib 18.6 released: Make your own object detector! Retrieved May 5, 2018, from http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html

King, D. E. (n.d.-c). dlib C++ Library: Easily Create High Quality Object Detectors with Deep Learning. Retrieved May 5, 2018, from http://blog.dlib.net/2016/10/easily-create-high-quality-object.html

King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, *10*, 1755–1758.

King, D. E. (2015). Max-Margin Object Detection. Retrieved from http://arxiv.org/abs/1502.00046

Köstinger, M., Wohlhart, P., Roth, P. M., & Bischof, H. (2011). Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. *Proceedings of the IEEE International Conference on Computer Vision*, 2144–2151. https://doi.org/10.1109/ICCVW.2011.6130513

Li, Haoxiang and Lin, Zhe and Shen, Xiaohui and Brandt, Jonathan and Hua, G. (2015). A Convolutional Neural Network Approach for Face Detection. *Cvpr*, 5325–5334. https://doi.org/10.1109/CVPR.2015.7299170

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9905 LNCS*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Martin~Abadi, Ashish~Agarwal, Paul~Barham, Eugene~Brevdo, Zhifeng~Chen, Craig~Citro, … Xiaoqiang~Zheng. (2015). {TensorFlow}: Large-Scale Machine Learning on Heterogeneous Systems. Retrieved from https://www.tensorflow.org/

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable Parallel Programming with CUDA. *Queue*, *6*(2), 40–53. https://doi.org/10.1145/1365490.1365500

NoScope: 1000x Faster Deep Learning Queries over Video · Stanford DAWN. (2017). Retrieved January 15, 2018, from http://dawn.cs.stanford.edu/2017/06/22/noscope/

NVIDIA TITAN X Graphics Card with Pascal | GeForce. (n.d.). Retrieved January 15, 2018, from https://www.geforce.co.uk/hardware/10series/titan-x/#redirected

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. Retrieved from http://arxiv.org/abs/1506.02640

Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. Retrieved from http://arxiv.org/abs/1612.08242

Shafiee, M. J., Chywl, B., Li, F., & Wong, A. (2017). Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video. Retrieved from http://arxiv.org/abs/1709.05943

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 1–14. https://doi.org/10.1016/j.infsof.2008.09.005

The Guardian view on surveillance in China: Big Brother is watching | Editorial | Opinion | The Guardian. (2017). Retrieved January 15, 2018, from https://www.theguardian.com/commentisfree/2017/dec/28/the-guardian-view-on-surveillance-in-china-big-brother-is-watching

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I-511-I-518. https://doi.org/10.1109/CVPR.2001.990517

Ward, D. (2014). Frame Rate Guide for Video Surveillance. Retrieved January 14, 2018, from https://ipvm.com/reports/frame-rate-surveillance-guide

Yang, S., Luo, P., Loy, C.-C., & Tang, X. (2016). WIDER FACE: A Face Detection Benchmark ,
. https://doi.org/10.1109/CVPR.2016.596

Zhang, K., Zhang, Z., Li, Z., Member, S., Qiao, Y., & Member, S. (2016). Joint Face Detection and Alignment using Multi - task Cascaded Convolutional Networks. *Spl*, (1), 1–5. https://doi.org/10.1109/LSP.2016.2603342

Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X., & Li, S. Z. (2017). S3FD: Single Shot Scale-Invariant Face Detector. *Proceedings of the IEEE International Conference on Computer Vision*, *2017–Octob*, 192–201. https://doi.org/10.1109/ICCV.2017.30